# Languages, Automata, Property Checking – Module A – Exercises

## 5XIE0 Computational Modeling

Twan Basten, Marc Geilen, Jeroen Voeten
Electronic Systems Group, Department of Electrical Engineering

# A.1 – languages

## 1 languages

Let $\Sigma$ be the alphabet $\{0, 1\}$

1. $L5 = \{\sigma \in \Sigma^* \mid (\forall i : i \in \mathbb{N} : \sigma(i) = 1 \Rightarrow \sigma(i+1) = 0)\}$
   00 ✓, 0100 ✓, 0110 ✗, 01010001 ✗ elements of $L5$ ?
2. $L6 = \{\sigma \in \Sigma^* \mid (\forall i : i \in \mathbb{N} : \sigma(i) = 1 \Rightarrow \sigma(i+1) = 1)\}$
   00 ✓, 0100 ✗, 0110 ✗, 01010001 ✗ elements of $L6$ ?
3. $L7 = \{\sigma \in \Sigma^* \mid (+i : i \in \mathbb{N} \wedge \sigma(i) = 1 : 1) \text{ is even}\}$
   $\varepsilon$ ✓, 00 ✓, 0100 ✗, 0110 ✓, and 01010001 ✗ elements of $L7$?
4. define $L8$ containing all words with an equal number of 1s and 0s.

$$L8 = \{\sigma \in \Sigma^* \mid (+i : i \in \mathbb{N} \wedge \sigma(i) = 0 : 1) = (+i : i \in \mathbb{N} \wedge \sigma(i) = 1 : 1)\}$$



# A.2.1 – regular languages

## 2 is language concatenation commutative?

no, for instance, $\{a\} \cdot \{b\} = \{ab\} \neq \{ba\} = \{b\} \cdot \{a\}$



## 3 understanding regular expressions

Recall $\alpha = 0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9$ , $\beta = \varepsilon + + + -$

- which of the following regular expressions also defines $L2 = (\beta\alpha\alpha^*(\varepsilon + E\alpha\alpha^*))$?
  - $\alpha_1 = \beta\alpha\alpha^*(E\alpha\alpha^*)$ ✗ • words in $L(\alpha_1)$ always contain an $E$ (precisely one)
  - $\alpha_2 = \beta\alpha\alpha^*(E\alpha\alpha)^*$ ✗ • $L(\alpha_2)$ allows multiple $E$s, and an $E$ is always followed by two digits
  - $\alpha_3 = \beta\alpha\alpha^*(E\alpha\alpha^*)^*$ ✗ • $L(\alpha_3)$ allows multiple $E$s
  - $\alpha_4 = (\beta\alpha\alpha^*(\emptyset + E\alpha\alpha^*))$ ✗ • $\emptyset + \rho = \rho$, so $L(\alpha_4) = L(\alpha_1)$

- which of these four expressions are define the same language?
  - only $\alpha_1$ and $\alpha_4$ – see above



## 3 understanding regular expressions

- which of the following pairs define the same language? counterexamples?
  - $\alpha = a(a^* + b^*),\ \beta = a(a + b)^*$
    - ✗ • $\alpha$ does not allow $a$-$b$ switching after initial $a$, whereas $\beta$ does
  - $\alpha = a(a^* + b)^*,\ \beta = a(a + b)^*$
    - $L(\alpha) = L(\beta)$; nested iteration in $\alpha$ does not add or remove options



## 4 creating regular expressions – L3

let $\beta = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9$; let $\alpha = 0 + \beta$
recall $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$

- $L3$ is the language of all numbers in which every $0$ is eventually followed by a $1$:

$$\{\sigma \in \Sigma^* \mid (\forall i : i \in \mathbb{N} : \sigma(i) = 0 \Rightarrow (\exists j : j \in \mathbb{N} \wedge j > i : \sigma(j) = 1))\}$$

e.g. $1201$ and $1400315$ are elements of $L3$, but $3210$ is not

  - $(\alpha^* 1)^* \beta^*$

## 4 creating regular expressions – L4

let $\beta = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9$; let $\alpha = 0 + \beta$
recall $\Sigma = \{0,1,2,3,4,5,6,7,8,9, +, =\}$

- $L4$ is the language of all natural numbers, well-formed additions and addition equations:
  - the word "0" is in $L4$
  - every nonempty word that does not contain "+" or "=" and does not start with "0" is in $L4$
  - a word containing "+" but not "=" separating two valid words of $L4$ is in $L4$
  - a word containing exactly one "=" and separating two valid words of $L4$ is in $L4$
  - no other word is in $L4$ other than those implied by the previous rules

e.g., 481, 27+15, 1+17 = 18, and 481+18=42 are elements of $L4$, but 040+2 is not

- $\gamma(+\gamma)^* + (\gamma(+\gamma)^* = \gamma(+\gamma)^*)$ with $\gamma = 0 + \beta\alpha^*$

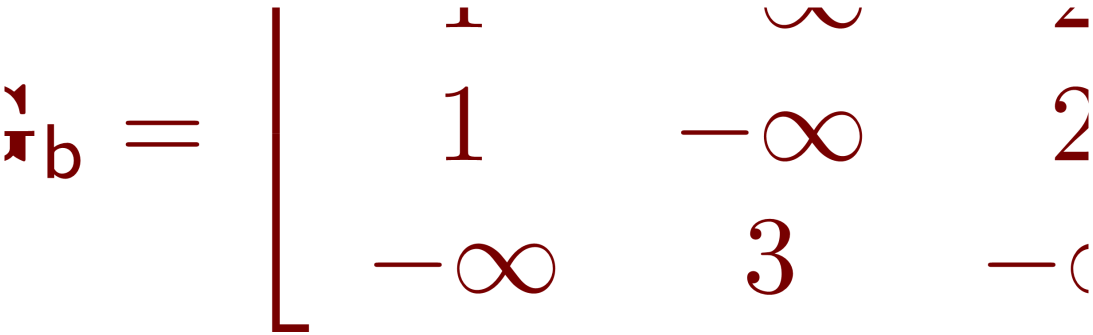


## 4 creating regular expressions – L5-L8

let $\Sigma = \{0,1\}$

- $L5 = \{\sigma \in \Sigma^* \mid (\forall i{:}\, i \in \mathbb{N}{:}\, \sigma(i) = 1 \Rightarrow \sigma(i+1) = 0)\}$
  - $(0 + 10)^*$
- $L6 = \{\sigma \in \Sigma^* \mid (\forall i{:}\, i \in \mathbb{N}{:}\, \sigma(i) = 1 \Rightarrow \sigma(i+1) = 1)\}$
  - $0^*$
- $L7 = \{\sigma \in \Sigma^* \mid (+i{:}\, i \in \mathbb{N} \wedge \sigma(i) = 1{:}\, 1) \text{ is even}\}$
  - $(0 + 10^*1)^*$
- $L8$ containing all words with an equal number of 1s and 0s.
  - $L8$ requires counting; this is not possible in regular languages

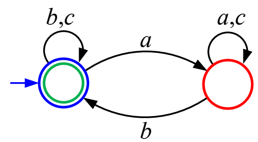




## A.2.2 – finite automata

## 5.1 languages of finite automata

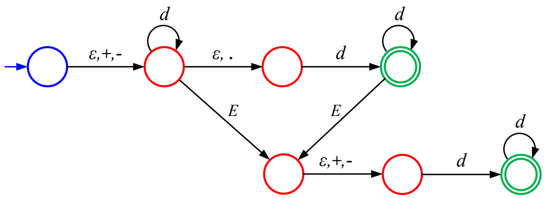
Automaton, with $\Sigma = \{a, b, c\}$



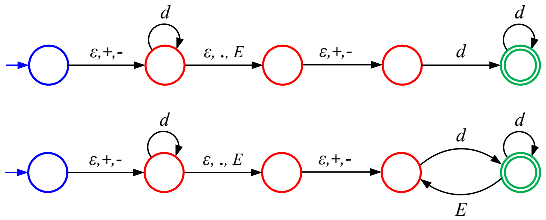

- every $a$ is always followed by a $b$
- $\{\sigma \in \Sigma^* \mid (\forall i{:}\, i \in \mathbb{N}{:}\, \sigma(i) = a \Rightarrow (\exists j{:}\, j \in \mathbb{N} \wedge j > i{:}\, \sigma(j) = b))\}$
- $(b + c + a(a + c)^*b)^*$

## 5.2 languages of finite automata

Numbers




Alternatives?



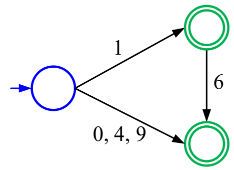

No
- both accept, for instance, .-42
- the second automaton accepts words with multiple $E$s



## 6.1 creating automata models

$L1 = \{0,1,4,9,16\}$ with $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$

DFA

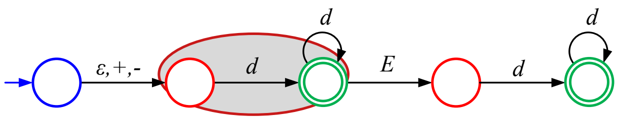




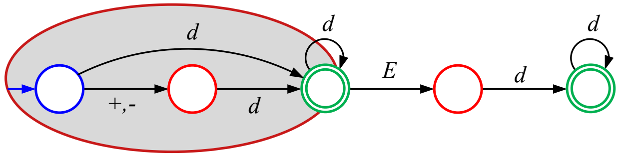

## 6.1 creating automata models

$L2$, the language of all integers of the form $< [+/-]\ number\ [E\ number] >$ with $\Sigma = \{0,1,2,3,4,5,6,7,8,9,+,-,E\}$

NFA




DFA





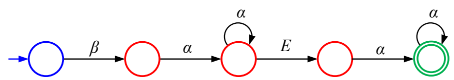

## 6.2 creating automata models

Recall $\alpha = 0+1+2+3+4+5+6+7+8+9$ , $\beta = \varepsilon + + + -$

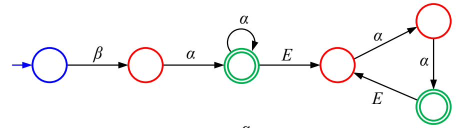
all nondeterministic!
(because of the $\varepsilon$ move in the initial state)

Automata?

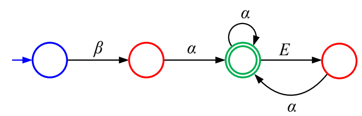
- $\alpha_1 = \beta\alpha\alpha^*(E\alpha\alpha^*)$
- $\alpha_4 = (\beta\alpha\alpha^*(\emptyset + E\alpha\alpha^*))$

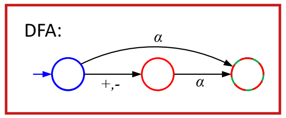



- $\alpha_2 = \beta\alpha\alpha^*(E\alpha\alpha)^*$




- $\alpha_3 = \beta\alpha\alpha^*(E\alpha\alpha^*)^*$



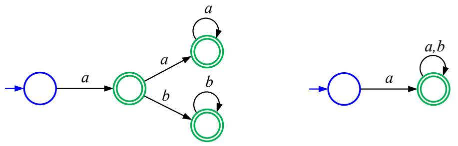



# 6.2 creating automata models

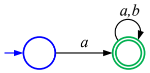
- Automata?

$\boxed{\text{all deterministic!}}$

  - $\alpha = a(a^* + b^*),\ \beta = a(a+b)^*$
    - $\alpha$ does not allow $a$-$b$ switching after initial $a$, whereas $\beta$ does

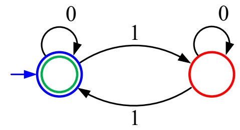



  - $\alpha = a(a^* + b)^*,\ \beta = a(a+b)^*$
    - $L(\alpha) = L(\beta)$; nested iteration in $\alpha$ does not add or remove options






# 6.3 creating automata models – L7




definition: $L7 = \{\sigma \in \Sigma^* \mid (+i{:}\, i \in \mathbb{N} \wedge \sigma(i) = 1{:}\, 1) \text{ is even}\}$; automaton?

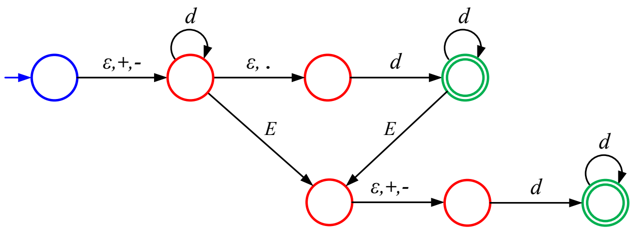
do it yourself, and check your answer with the workbench

$\boxed{(0 + 10^*1)^*}$

- enter your automaton YA in the workbench
- take the regular expression of Exercise 4 and enter it in the workbench
- convert the expression to a finite automaton EA
- check language inclusion $L(\text{YA}) \subseteq L(\text{EA})$ between your automaton and the expression automaton
- check language inclusion $L(\text{EA}) \subseteq L(\text{YA})$ between the expression automaton and your automaton
- if both inclusions hold, then $L(\text{EA}) = L(\text{YA})$ and hence your answer is correct; if any of the inclusions does not hold, then learn from the counterexamples and retry

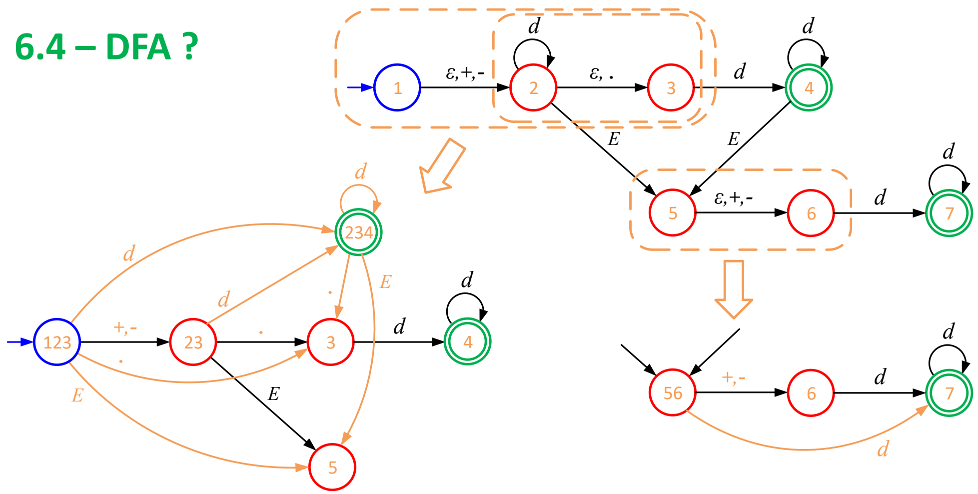


# 6.4 – DFA ?



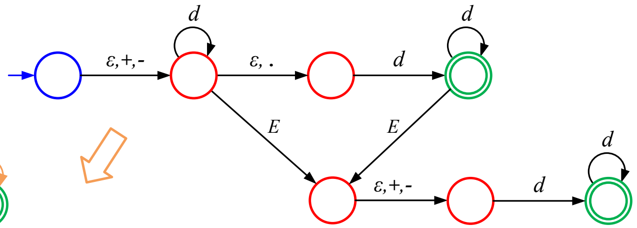


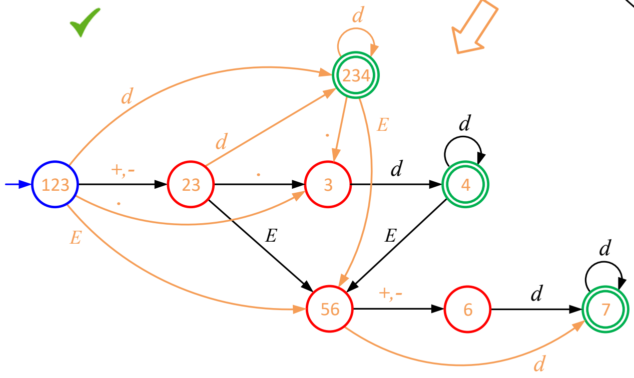

# 6.4 – DFA ?



## 6.4 – DFA ?

workbench – **check !**






## 6.5 creating automata models – L8

$L8 = \{\sigma \in \Sigma^* \mid (+i{:}\, i \in \mathbb{N} \wedge \sigma(i) = 0{:}\, 1) = (+i{:}\, i \in \mathbb{N} \wedge \sigma(i) = 1{:}\, 1)\}$

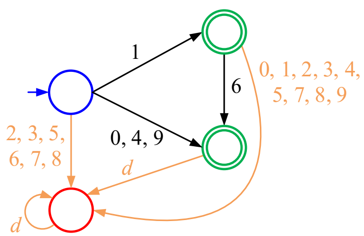
no NFA can be created

keeping track of all possible differences between 0-counts and 1-counts cannot be done with finitely many states



## 7.1 completing automata models – L1

$L1 = \{0,1,4,9,16\}$ with $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$

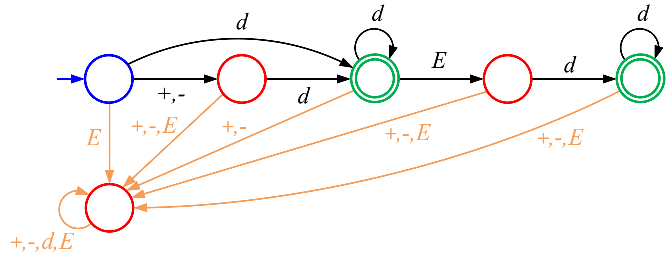





## 7.1 completing automata models – L2

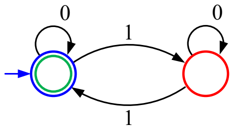
$L2$, the language of all integers of the form $< [+/-]\ number\ [E\ number] >$

with $\Sigma = \{0,1,2,3,4,5,6,7,8,9,+,-,E\}$



## 7.1 completing automata models – L7

- definition: $L7 = \{\sigma \in \Sigma^* \mid (+i : i \in \mathbb{N} \wedge \sigma(i) = 1 : 1) \text{ is even}\}$
- regular expression: $(0 + 10^*1)^*$
- automaton




– a complete dfa

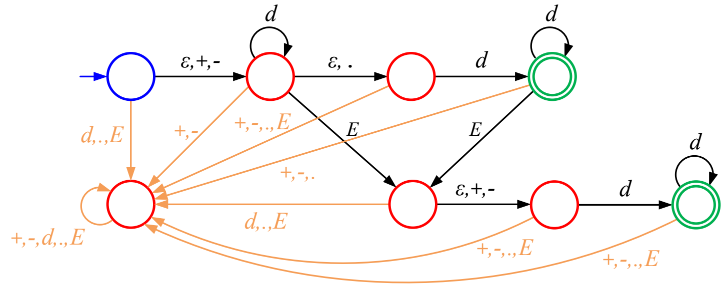


## 7.2 completing automata models

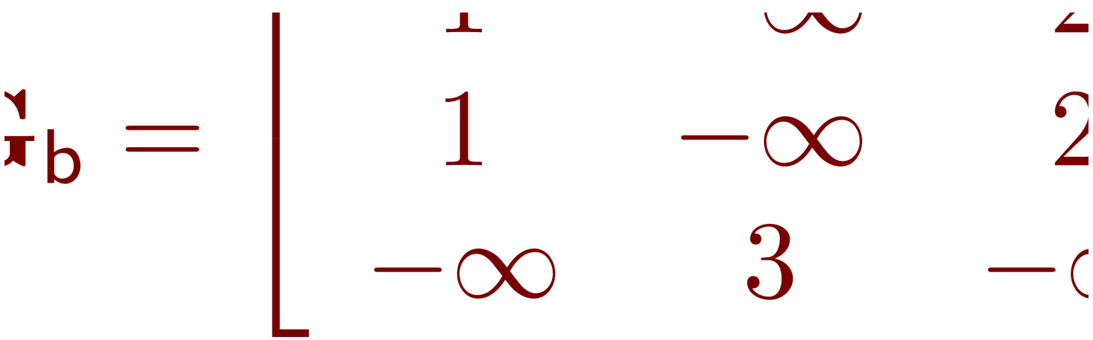
6.7 and +4.5

two words with

- only one accepting run in the original NFA
- an additional non-accepting run in the completed NFA ?



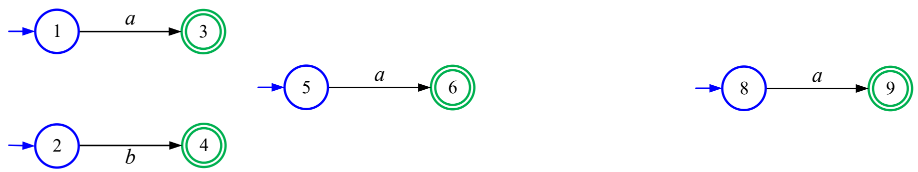


## A.2.4 – conversions between representations of regular languages

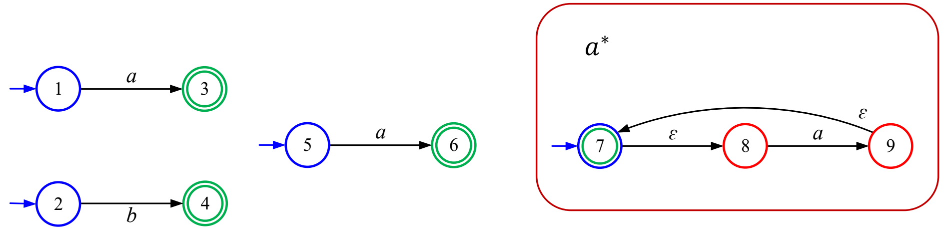
## 11 from regular expression to NFA-ε

$$(a + b)^* a a^*$$



# 11 from regular expression to NFA-ε

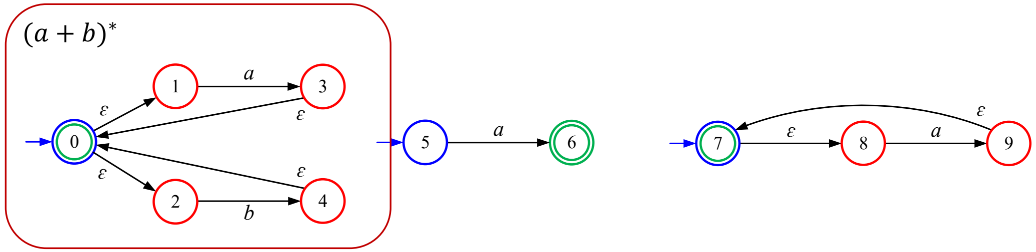
$$(a+b)^*aa^*$$






# 11 from regular expression to NFA-ε

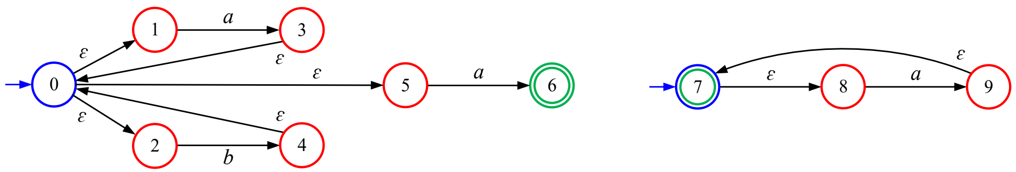
$$(a+b)^*aa^*$$






# 11 from regular expression to NFA-ε

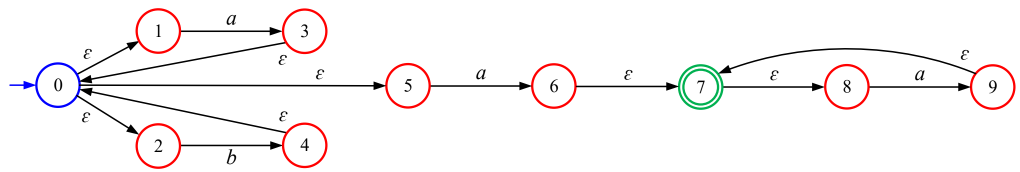
$$(a+b)^*aa^*$$






# 11 from regular expression to NFA-ε

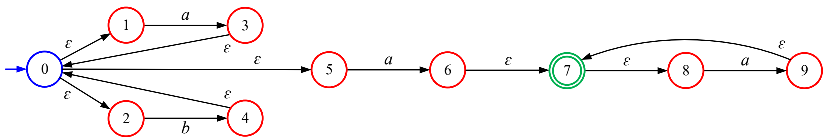
$$(a+b)^*aa^*$$

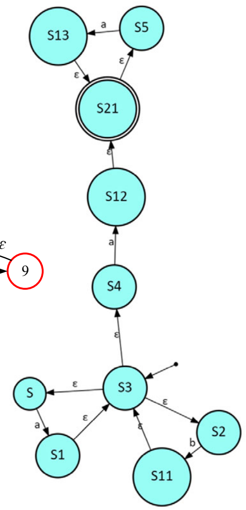


# 11 from regular expression to NFA-ε

workbench – **check !**

$(a+b)^* aa^*$

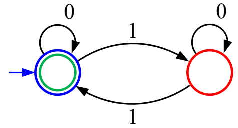




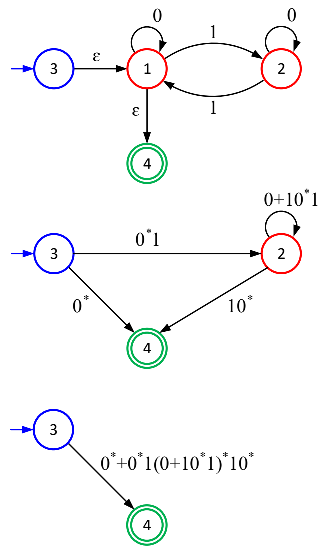

# 12.1 NFA 2 re – state elimination – L7

- definition: $L7 = \{\sigma \in \Sigma^* \mid (+i : i \in \mathbb{N} \wedge \sigma(i) = 1 : 1) \text{ is even}\}$
- regular expression: $(0 + 10^*1)^*$
- automaton – a complete dfa:




- workbench – **check !**

$$'0'^* \cdot '1' \cdot \left('0' + '1' \cdot '0'^* \cdot '1'\right)^* \cdot '1' \cdot '0'^* + '0'^*$$

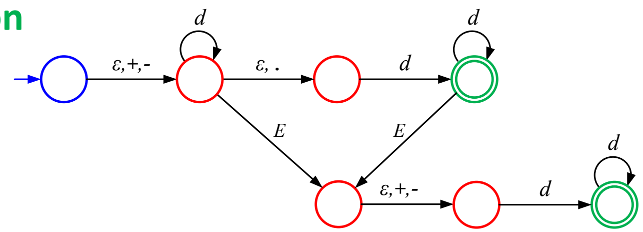


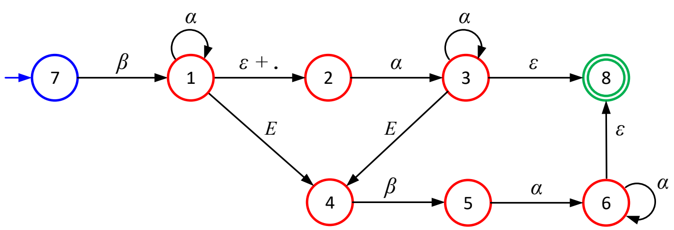



# 12.2 state elimination

$\alpha = 0 + 1 + \cdots + 8 + 9$

$\beta = \varepsilon + + + -$



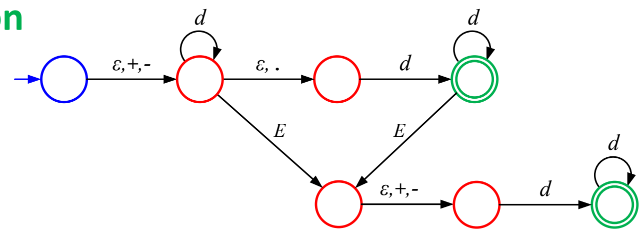

right format, initialization

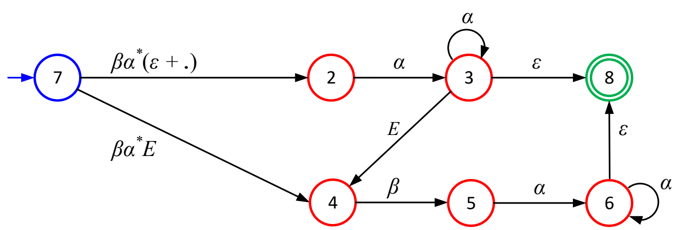


# 12.2 state elimination

$\alpha = 0 + 1 + \cdots + 8 + 9$

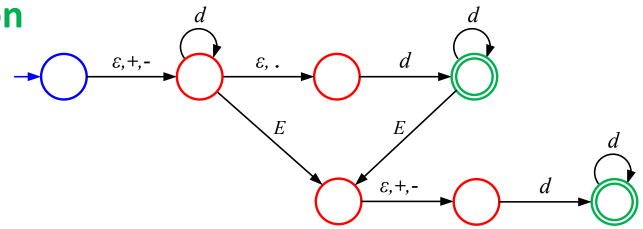
$\beta = \varepsilon + + + -$



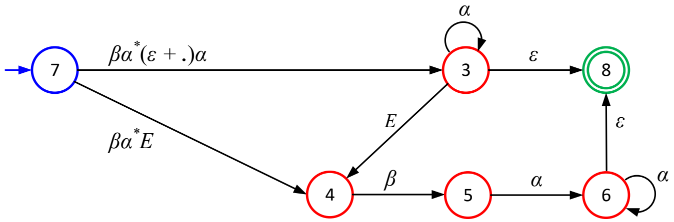




eliminate 1

## 12.2 state elimination



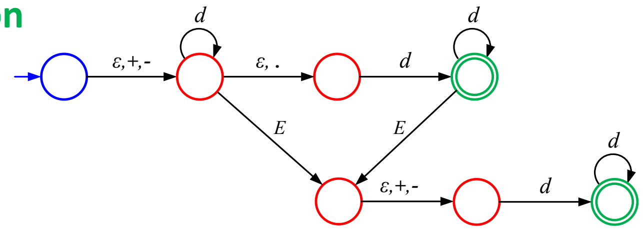

$$\alpha = 0 + 1 + \cdots + 8 + 9$$
$$\beta = \varepsilon + \mathbf{+} + \mathbf{-}$$



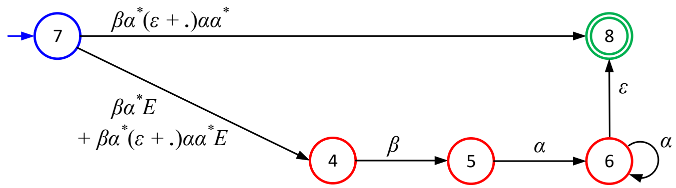

eliminate 2



## 12.2 state elimination



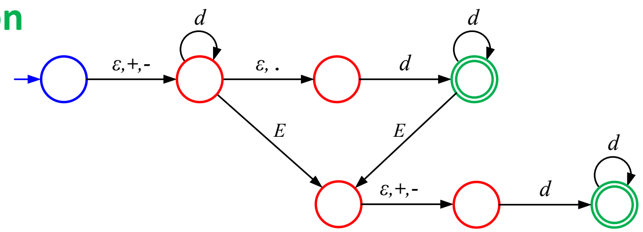

$$\alpha = 0 + 1 + \cdots + 8 + 9$$
$$\beta = \varepsilon + \mathbf{+} + \mathbf{-}$$



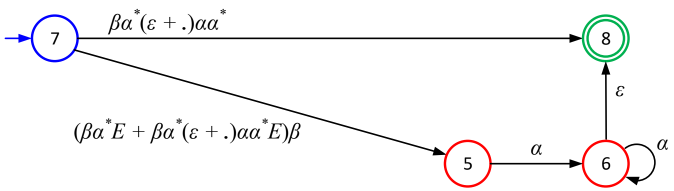

eliminate 3



## 12.2 state elimination



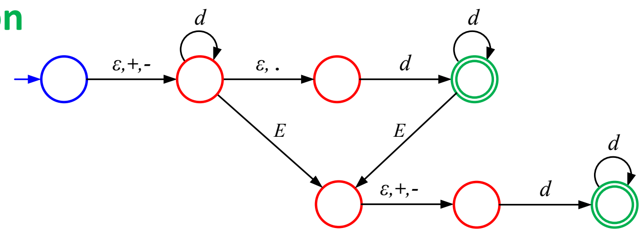

$$\alpha = 0 + 1 + \cdots + 8 + 9$$
$$\beta = \varepsilon + \mathbf{+} + \mathbf{-}$$



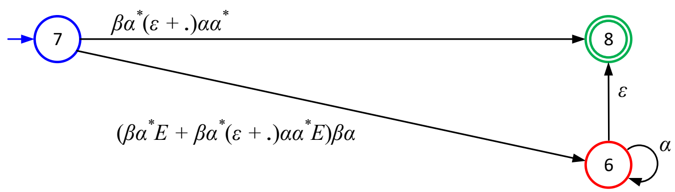

eliminate 4



## 12.2 state elimination

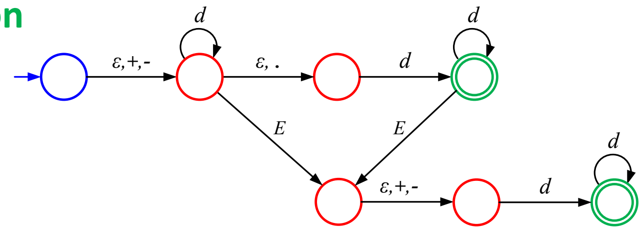



$$\alpha = 0 + 1 + \cdots + 8 + 9$$
$$\beta = \varepsilon + \mathbf{+} + \mathbf{-}$$



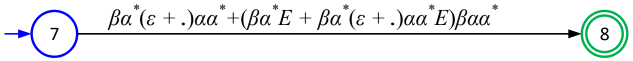

eliminate 5

## 12.2 state elimination




$\alpha = 0 + 1 + \cdots + 8 + 9$

$\beta = \varepsilon + \mathbf{+} + -$



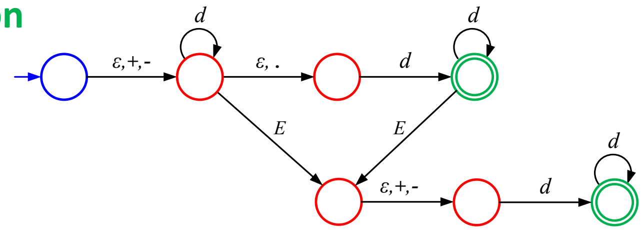

eliminate 6

end result

$\beta\alpha^*(\varepsilon + .)\alpha\alpha^* + (\beta\alpha^* E \;+\; \beta\alpha^*(\varepsilon + .)\alpha\alpha^* E)\beta\alpha\alpha^*$



## 12.2 state elimination




$\alpha = 0 + 1 + \cdots + 8 + 9$

$\beta = \varepsilon + \mathbf{+} + -$

workbench – **check !**

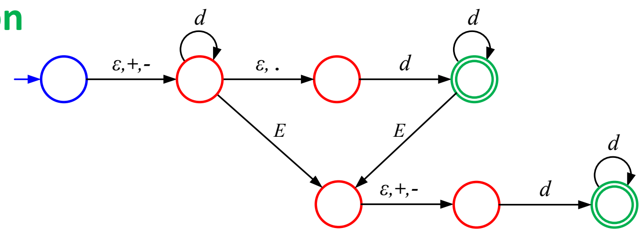
$$('-' + '+' + \epsilon) \cdot ('0' + '6' + '3' + '7' + '4' + '9' + '1' + '8' + '5' + '2')^* \cdot (\epsilon + '.')$$

end result

$\beta\alpha^*(\varepsilon + .)\alpha\alpha^* + (\beta\alpha^* E \;+\; \beta\alpha^*(\varepsilon + .)\alpha\alpha^* E)\beta\alpha\alpha^*$



## 12.2 state elimination




$\alpha = 0 + 1 + \cdots + 8 + 9$

$\beta = \varepsilon + \mathbf{+} + -$

workbench – **check !** ✓

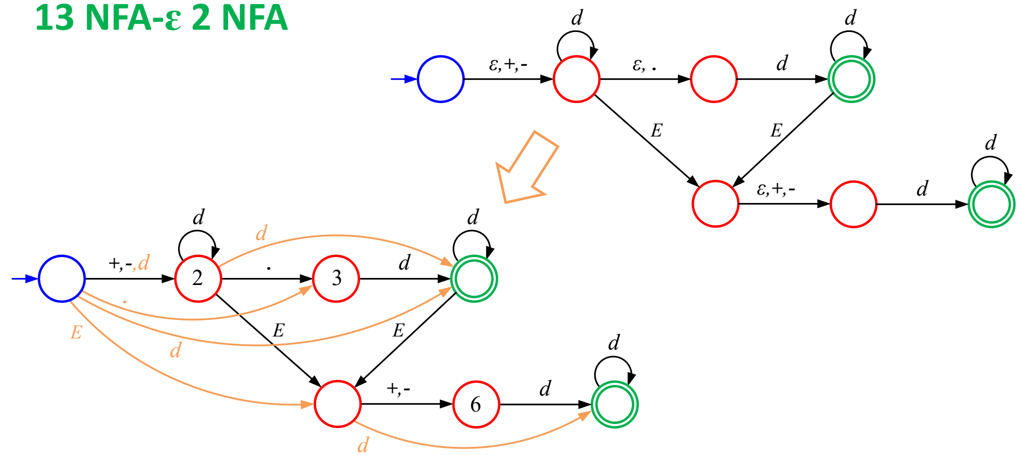
$$b \cdot a^* \cdot ('.' + \epsilon) \cdot a \cdot a^* + (b \cdot a^* \cdot ('.' + \epsilon) \cdot a \cdot a^* \cdot E \cdot b \cdot a + b \cdot a^* \cdot E \cdot b \cdot a) \cdot a^*$$

end result

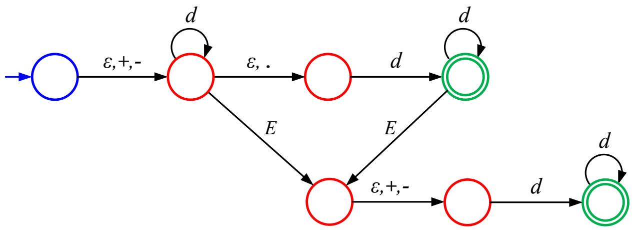
$\beta\alpha^*(\varepsilon + .)\alpha\alpha^* + (\beta\alpha^* E \;+\; \beta\alpha^*(\varepsilon + .)\alpha\alpha^* E)\beta\alpha\alpha^*$



## 13 NFA-ε 2 NFA



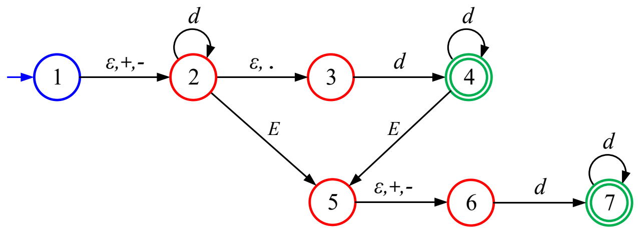
# 14.4 NFA-ε 2 DFA






ES ELECTRONIC SYSTEMS TU/e

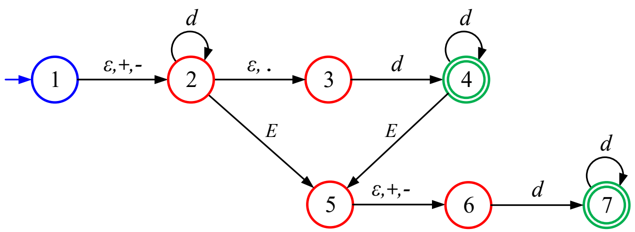
# 14.4 NFA-ε 2 DFA

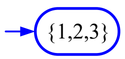





ES ELECTRONIC SYSTEMS TU/e

# 14.4 NFA-ε 2 DFA

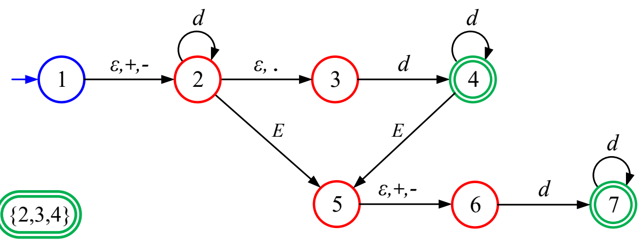


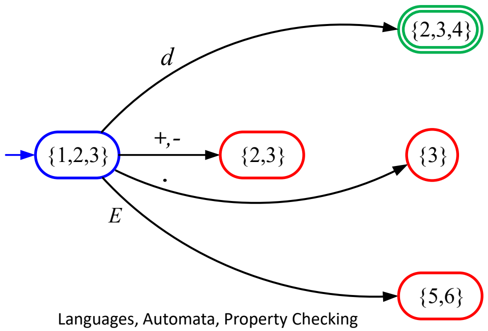






ES ELECTRONIC SYSTEMS TU/e

# 14.4 NFA-ε 2 DFA

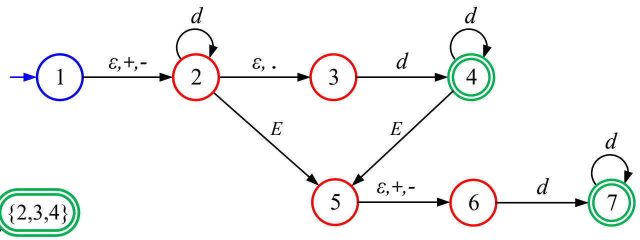


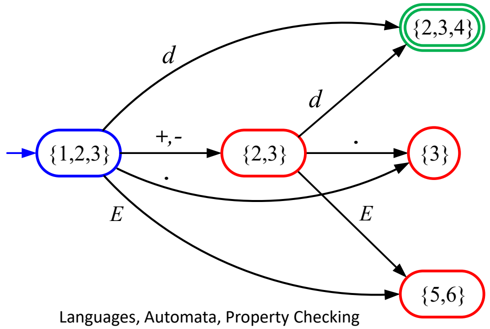






ES ELECTRONIC SYSTEMS TU/e

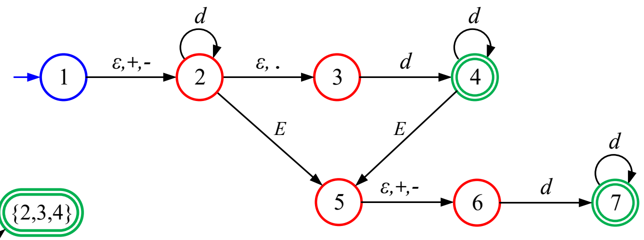
## 14.4 NFA-ε 2 DFA

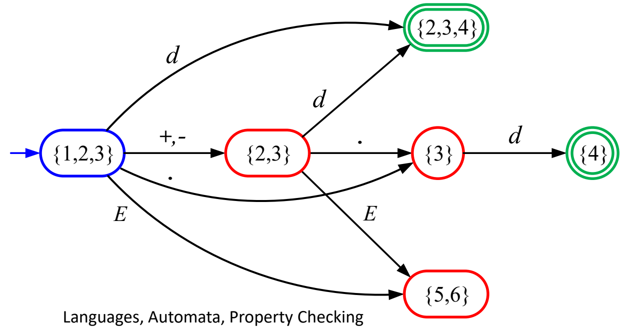




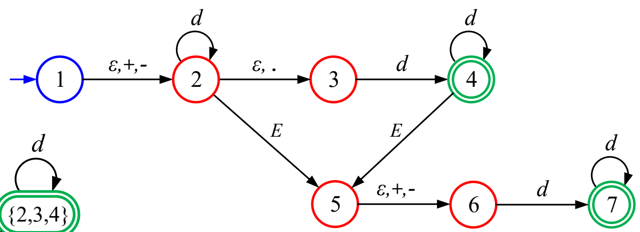

## 14.4 NFA-ε 2 DFA

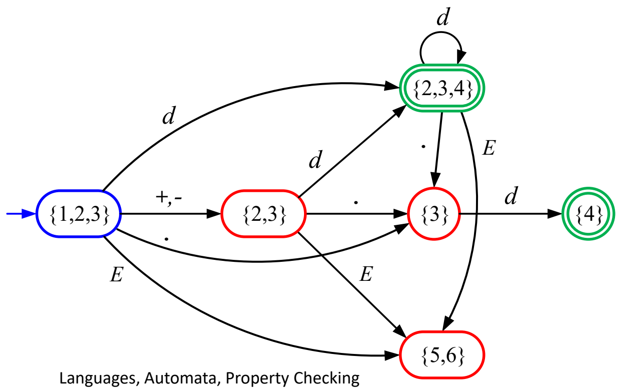




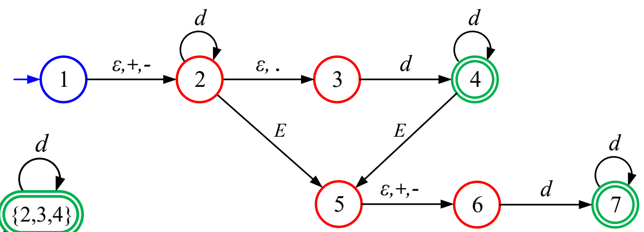

## 14.4 NFA-ε 2 DFA

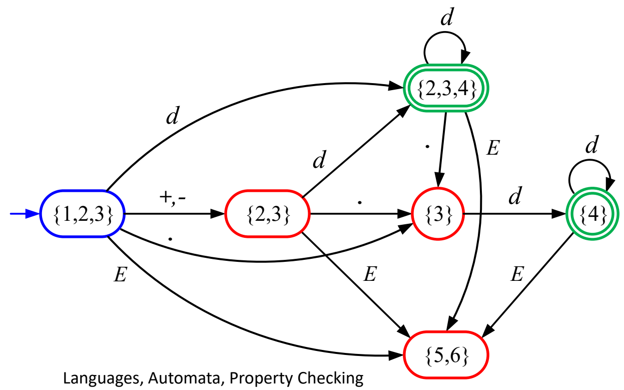


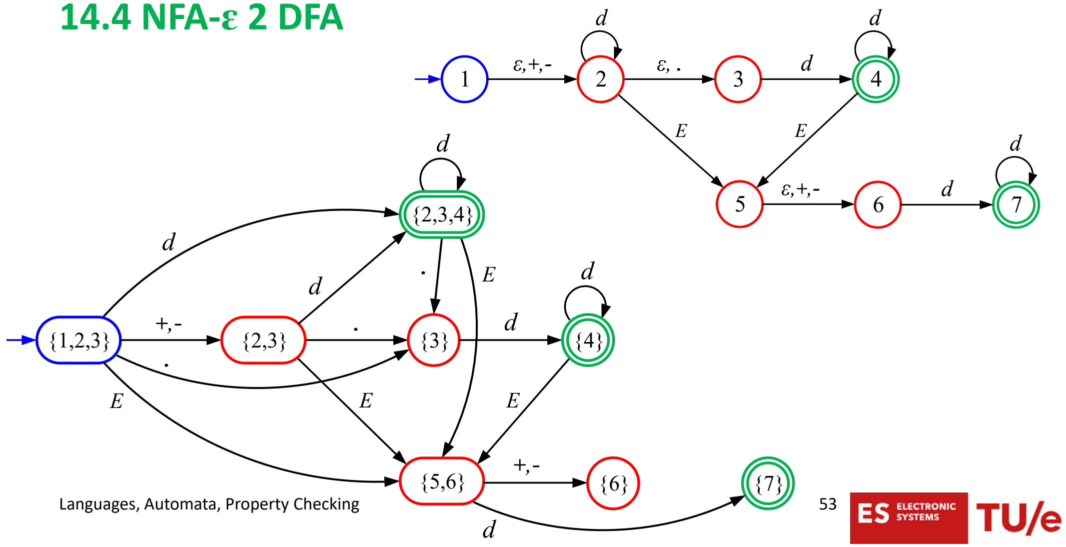



## 14.4 NFA-ε 2 DFA



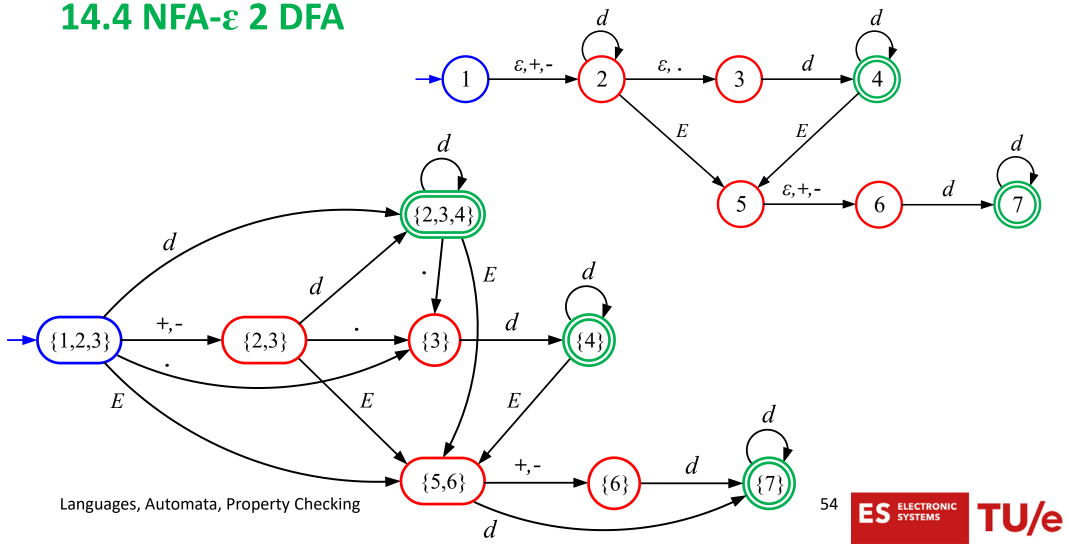



## 14.4 NFA-ε 2 DFA

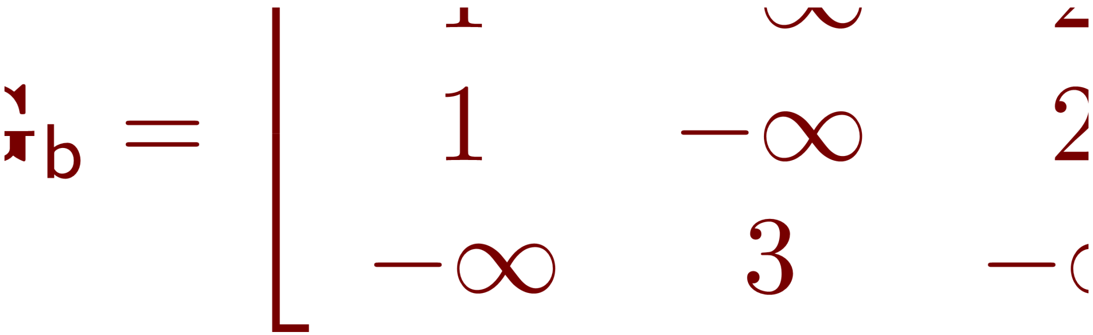





## 14.4 NFA-ε 2 DFA



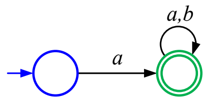


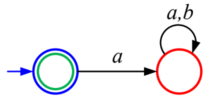
## A.2.3 – expressiveness

## 8.1 complementing incomplete DFA

- incomplete DFA $A$ with $L(A) = \{\sigma | \sigma(0) = a\}$

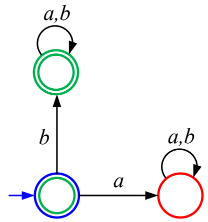



- complement $\bar{A}$ – swapping (non-)final states




- $L(\bar{A}) = \{\varepsilon\} \neq \overline{L(A)}$

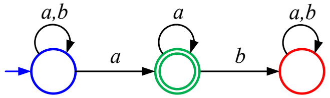
- complete and complement, $\overline{cA}$



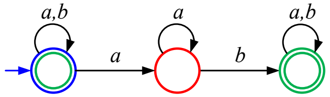

- $L(\overline{cA}) = \overline{L(A)} = \{\varepsilon\} \cup \{\sigma | \sigma(0) = b\}$

## 8.2 complementing complete NFA

- complete NFA $cA$ with $L(cA)$ containing all words ending with an a




- complement $\overline{cA}$ – swapping (non-)final states




- $L(\overline{cA}) = \Sigma^* \neq \overline{L(cA)}$



## 9.1 pumping lemma – L2

- $L2$ is the language of all integers of the form $< [+/-]\ number\ [E\ number] >$
- Recall the following regular expression for $L2$ :

  $\beta\alpha\alpha^*(\varepsilon + E\alpha\alpha^*)$ with $\alpha = 0+1+2+3+4+5+6+7+8+9$ , $\beta = \varepsilon + + + -$
- $5$ is a valid pumping length
- any word $w$ in $L2$ with $|w| \geq 5$ has a two-digit subword, of which one digit can be dropped/pumped
- example $+1E23$; subwords that can be pumped are $2$ and $3$; $+1E3$, $+1E2223$ are elements of $L2$
- example $456E7$; subwords that can be pumped are $4, 5, 6, 45$ and $56$; $6E7$, $45456E7$ are in $L2$

- a smaller pumping length does not exist
- e.g., $+1E2$ does not have a part that can be repeated arbitrarily often, including zero (!) times



## 9.2/3 proving non-regularity using the pumping lemma

- 9.2: $L8$ – equal number of 0s and 1s
- 9.3: nested pairs of braces {{…}{}}…

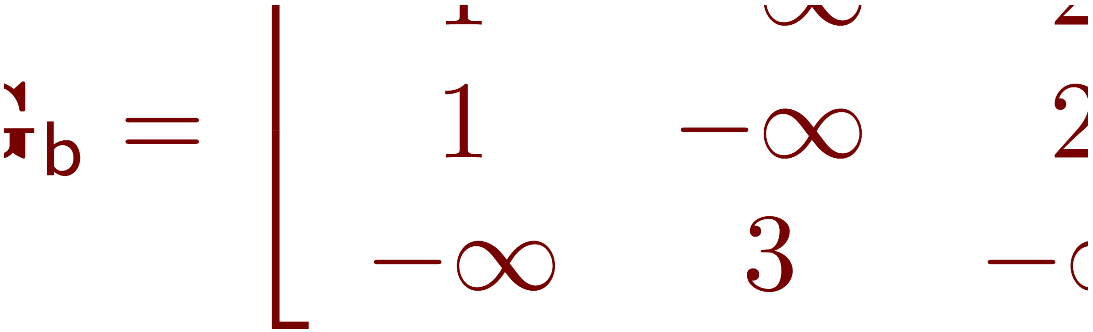
Proving non-regularity of these languages follows Example A.8 (quite literally … )

- Towards a contradiction, assume $p$ is a valid pumping length
- Then $\sigma = a^p b^p$ (with $a = 0, b = 1$ resp. $a = \{, b = \}$) is an element of the language
- But $|\sigma| > p$ and dropping or repeating any part of $a^p$ will not give a word in the language
- Hence, $p$ cannot exist and the languages are not regular



## 10 proving non-regularity using the pigeonhole principle

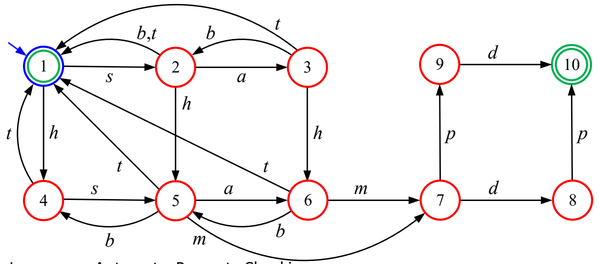
reasoning follows Example A.9 with $a = 0, b = 1$ resp. $a = \{, b = \}$



## A.3 – property checking

## 15 checking regular properties

- Property: $\alpha^*(\varepsilon + m\alpha^*(d\alpha^*p + p\alpha^*d)\alpha^*)$ with $\alpha = s + a + b + h + t$



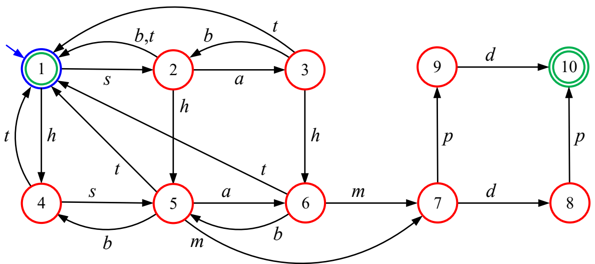

*s – select drink*
*a – adapt settings*
*b – back*
*h – hold card*
*t – time out*
*m – make drink*
*d – deliver drink*
*p – (execute) payment*



## 15 checking regular properties

- Property: $\alpha^*(\varepsilon + m\alpha^*(d\alpha^*p + p\alpha^*d)\alpha^*)$ with $\alpha = s + a + b + h + t$
- CMWB:

```
regular expression Prop = @alpha*.(\e+m.@alpha*.(d.@alpha*.p+p.@alpha*.d).@alpha*)
where alpha = s+a+b+h+t
```

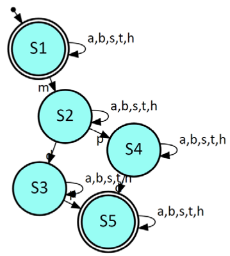


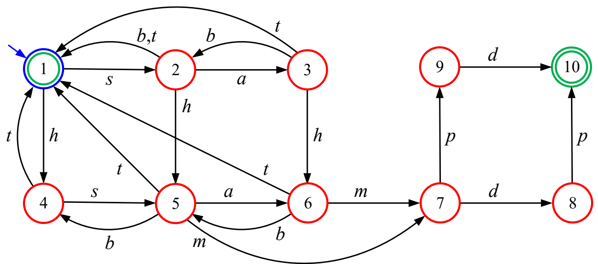
*s – select drink*
*a – adapt settings*
*b – back*
*h – hold card*
*t – time out*
*m – make drink*
*d – deliver drink*
*p – (execute) payment*



## 15 checking regular properties

- 2 NFA, DFA, minimize, relable

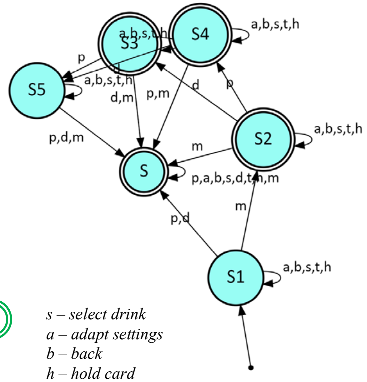


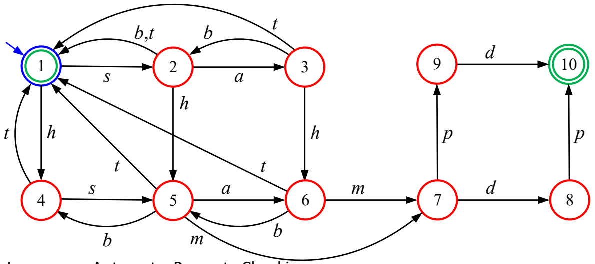



*s – select drink*
*a – adapt settings*
*b – back*
*h – hold card*
*t – time out*
*m – make drink*
*d – deliver drink*
*p – (execute) payment*

# 15 checking regular properties

- complete, complement



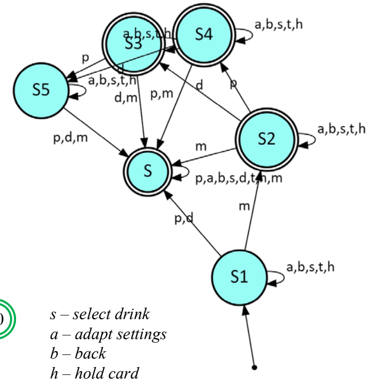



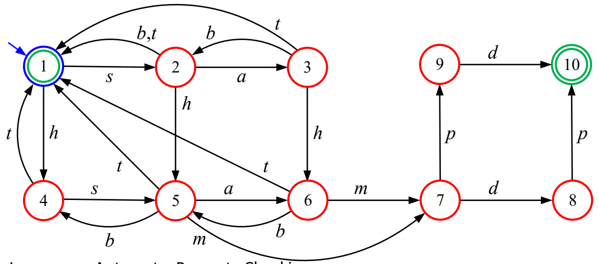

*s – select drink*
*a – adapt settings*
*b – back*
*h – hold card*
*t – time out*
*m – make drink*
*d – deliver drink*
*p – (execute) payment*



# 15 checking regular properties

- product



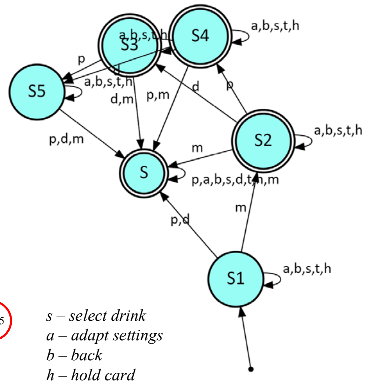


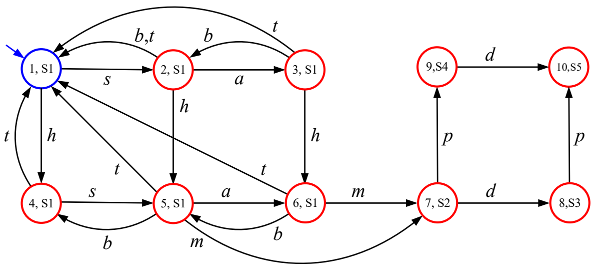
*s – select drink*
*a – adapt settings*
*b – back*
*h – hold card*
*t – time out*
*m – make drink*
*d – deliver drink*
*p – (execute) payment*



# 15 checking regular properties

- product



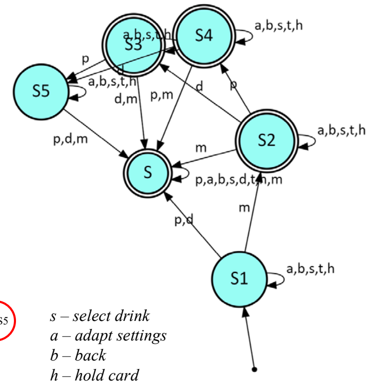


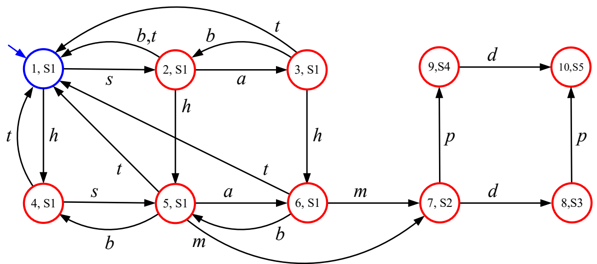

*s – select drink*
*a – adapt settings*
*b – back*
*h – hold card*
*t – time out*
*m – make drink*
*d – deliver drink*
*p – (execute) payment*



# 15 checking regular properties

- emptiness check: ✓






*s – select drink*
*a – adapt settings*
*b – back*
*h – hold card*
*t – time out*
*m – make drink*
*d – deliver drink*
*p – (execute) payment*